DOCUMENT RESUME

ED 188 605 IR 008 467

AUTHOR Rothstein, Jerome
TITLE Parallel Processable Cryptographic Methods with Unbounded Practical Security.
INSTITUTION Ohio State Univ., Columbus. Computer and Information Science Research Center.
REPORT NO TR-OSU-CISRC-77-9
PUB DATE Jul 77
NOTE 18p.: Paper presented at the IEEE International Symposium on Information Theory (Ithaca, NY, October 10-13, 1977).

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Algorithms: *Codification: Computer Programs: *Confidentiality: *Data Bases: Information Theory: Online Systems

ABSTRACT

Addressing the problem of protecting confidential information and data stored in computer databases from access by unauthorized parties, this paper details coding schemes which present such astronomical work factors to potential code breakers that security breaches are hopeless in any practical sense. Two procedures which can be used to encode for reliability and noise immunity are described: groupoid and quasigroupoid string coding systems, and bus automata used as parallel coders and decoders. References related to cryptographic security are listed. (FM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED188605

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

OSU-CISRC-TR-77-9

PARALLEL PROCESSABLE CRYPTOGRAPHIC METHODS WITH UNBOUNDED PRACTICAL SECURITY

BY

Jerome Rothstein

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Celianne Taylor

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Computer and Information Science Research Center

The Ohio State University

Columbus, Ohio 43210

July, 1977

IR008467

TABLE OF CONTENTS

	PAGE
ABSTRACT	iii
I. Introduction	1
II. Groupoid and Quasigroup String Coding Systems	3
III. Bus Automata as Parallel Coders and Decoders	8
IV. Conclusion	11

PREFACE

This report is the result of research supported by the Department of Computer and Information Science and the Computer and Information Science Research Center (CISRC) of the Ohio State University. A presentation of the research results is being presented at the IEEE International Symposium on Information Theory, Ithaca, N. Y., October 10-13, 1977.

Parallel Processing Cryptographic Methods with Unbounded Practical Security

Jerome Rothstein
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

Cryptographic methods are described which, at reasonable cost to users, impose prohibitive work penalties on code-breakers. The class of cryptographic transformations is so vast that codes can be used once and discarded, both in communication and in data security applications. We use the groupoid string formalism (earlier shown to combine Turing universality with parallel computing capability) specialized to quasigroups for unique decodability. Coding for error detection and correction can be done compatibly and independently. It appears likely that crypto and reliability encoding can be systematically combined to decoy code-breakers into semanticizing their decoding of reliability procedures into message opacifiers. Identification, authorization, and handshaking procedures can be integrated into the cryptographic method (with total update capability) at incremental user cost comparable to separate cost of those procedures or less.

I. Introduction

The growth of large computer data banks containing confidential information has spurred development of security measures of many types. They include access control via proper identification and authorization, variation in the kinds of access permitted, and other adaptations of the gamut of security measures familiar to the military, including cryptography. The data protection problem differs from the communication protection problem in several respects. Many users, sharing the same computer facility, need both to cooperate in some areas and to be protected against each other in others. The data to be protected is often stored for long times rather than having an ephemeral existence. Recording, if a security breach is suspected, may entail unacceptable overhead cost not encountered in communication cryptography. The "enemy" has access to the capability of the computer to break codes for access to and decryption of data stored by that and/or other computers. From the user's viewpoint, employees of an outside computer utility may have to be regarded as potential enemies. For these and other reasons it is justified to assume that the user's total encrypted data base is available to the enemy. This makes the first of two secrecy system parameters discussed by Shannon, namely unicity distance and work factor, essentially irrelevant.

The object of this paper is to develop coding schemes which present such astronomical work factors to potential enemies that code-breaking is hopeless in any practical sense, and to do so at acceptable cost to the user. As will be seen, the procedures involved are well adapted to parallel computation. Indeed, their discovery is a by-product of research in cellular automata and parallel computation. The cryptographic code or key space is made so large, combinatorially, that even if every single use by every single user employed a key which is never used again, only a negligeably

small fraction of the usable keys are "consumed". For orientation, note that there are about 3×10^{13} microseconds in a year; the number of different codes can easily be made much greater than 10 raised to this number. Indeed, this process of exponentiation can be iterated with no visible limit to this other than increased overhead associated with delays in encoding/decoding (for fixed computer capability) entailed by greated computational complexity.

Encoding for reliability and/or noise immunity is possible by two independent but compatible kinds of procedure. The usual coding theory methods, conventionally employed to combat noise and utilize channel capacity efficiently, can always be invoked to protect data already encoded by the algorithmic methods of main interest in this paper. Indeed, the statistical properties of messages encoded for noise immunity alone enable code-breakers to discover them and use them to decipher the protected message. This suggests using reliability codes of a complex nature, say alternating several in such a way that cracking the system determining the alternation scheme yields a meaningful, but bogus message (or a misleading one). When the bogus clear message emerges, the operations performed to make this occur will thoroughly scramble the real encoded message. The legitimate user, of course, merely "peels off" the reliability encoding and decodes in earnest.

The second class of methods utilizes error detection and correction capabilities inherent in the encoding/decoding procedures themselves. As will be seen, these can be expanded and diversified in systematic ways and generalize to include a variety of security measures. Among them are identification, authorization, handshaking, updating, and decoy capabilities like those of the first class.

II. Groupoid and Quasigroup String Coding Systems

In this section we first show how groupoid string systems, invented to investigate algorithmic pattern generation [1], and later shown to have computation universality [2], leads naturally to encoding systems executable in parallel by long shift registers or bus automata. We then introduce quasigroups for unique decodability and indicate the astronomical numbers of distinct encoding/decoding algorithms available.

Let G be a finite set of symbols used to form strings (words, messages) by concatenation in the usual way, which is also made into a groupoid (G,*) by a multiplication table for the binary operation *, under which G is closed. From a given string π over G

$$\pi = \cdots p_i \, p_{i+1} \cdots \tag{2.1}$$

we compute its daughter δ

$$\delta = \cdots d_i \, d_{i+1} \cdots \tag{2.2}$$

by $$d_i = p_i * p_{i+1} \tag{2.3}$$

or equivalently, by

$$d_i = p_{i-1} * p_i \tag{2.3'}$$

For example, the infinite periodic binary string (under $\oplus$) with period (0111) has as its daughter the string with period (10001). The granddaughter is (10010), while the great granddaughter is the original string. In general, the n^{th} descendant symbols are complex functions of (n+1) contiguous symbols of the parent string. Using d_i^k to represent the i^{th} symbol of the k^{th} generation descendant we have, recursively,

$$\begin{aligned} d_i^1 &= p_i * p_{i+1} \\ d_i^2 &= (p_i * p_{i+1}) * (p_{i+1} * p_{i+2}) \\ d_i^3 &= ((p_i * p_{i+1}) * (p_{i+1} * p_{i+2})) * ((p_{i+1} * p_{i+2}) * (p_{i+2} * p_{i+3})) \\ &\vdots \\ d_i^n &= d_i^{n-1} * d_{i+1}^{n-1} \end{aligned} \tag{2.4}$$

The increasing complexity is readily seen; the (n+1) elements involved in d_i^n are combined by (2^n-1) *-operations, and for non-associative groupoids the number of pairs of parentheses is 2^n-2. If G is both associative and commutative, parentheses can be dropped and *-powers of elements collected as in the derivation of the bionomial theorem. In this simple case we have

$$d_i^n = p_i^{\binom{n}{0}} * p_{i+1}^{\binom{n}{1}} * p_{i+1}^{\binom{n}{2}} * \ldots * p_{i+n}^{\binom{n}{n}} \tag{2.5}$$

Note that computation of d_i^n can be done in parallel with shift register logic appropriate to *-multiplication. The binary operation of conventional shift registers is, of course no problem. If m binary digits are needed to encode all the groupoid elements, the *-shifts become left shifts of m cells. The logical functions are then given by the mapping of Boolean m-vectors

$$B^m \times B^m \to B^m \tag{2.6}$$

corresponding to the *-multiplication table; they are calculated by standard logical design techniques.

There are $k^{(k^2)}$ distinct groupoids of order k; for k = 20 this number exceeds 10^{520}. Many groupoids are useless for our purposes in that encodings in terms of descendants may not be decodable to regain the parent. A binary example is the groupoid with only 0 entries in the table: all descendants of any string contain no symbols other than 0. We therefore restrict our attention to quasigroups, which are groupoids with unique two-sided solvability, i.e., if a * b = c, one can solve for any one of a, b, & c (from the *-table) given the other two. The number of quasigroups of order k is precisely L_k, the number of latin squares of order k.

Except for $k \leq 9$, only upper and lower bounds are known for L_k, the latter due to Hall [3], the former (not yet published) due to the writer. Writing

$$M_k = 1 \,!\, 2 \,!\, 3 \,!\, \ldots\, k \,!$$

we have, for $k > 2$,

$$M_k < L_k < M_k \; M_k - 1 \tag{2.7}$$

A loose lower bound for L_n due to Ryser [5] is

$$L_k < (e^{-2}k)^{k^2/2} \tag{2.8}$$

For orientation, the number of reduced latin squares of order 9 (i.e., top row and left column in standard order, multiply by 9 ! 8 ! to get L_9) was computed exactly by S. E. Bammel and the writer [4] and found to be 377,597,570,964,258,816. The number L_{32}, by (2.7), is astronomical and its symbols need only five bits for their encoding.

But even with vast L_k we have barely begun to tap the richness of our possible encodings: in each of n successive daughter-generating operations we are free to choose a different L_k for our multiplication table, i.e., $(L_k)^n$ different encoding procedures are usable. Furthermore, we can even choose a different L_k to compute successive elements of the same daughter. For m such choices we have $(L_k)^m$ different encodings or $(L_k)^{mn}$ ways for n^{th} generation strings. Here m can be message length or longer, n is unrestricted, and k, the number of symbols, can be made as large as desired with little penalty to the user. The number of encoding algorithms, C, thus can easily be made to satisfy

$$C > (ak)^{bmnk^2} \tag{2.9}$$

where a and b are constants, and m, n, k are integers which can be as large as desired.

To see that m, the effective length of the message can be made arbitrarily long, merely prefix a long string α and/or add a long string β at its end. These strings can be chosen to perform what is effectively an identification or authorization procedure, for the encoding procedure, via daughter formation, folds them algorithmically into the message.

Removal of their effect, at an appropriate stage of multistage decoding, would be possible for properly identified and authorized users (those knowing α and β) but effectively impossible without specific knowledge of α and β. We can even make m effectively infinite. For example, assign specific portions of α and β adjacent to the message being encoded, corresponding to α and β uses previously mentioned, and then let the string go to infinity in both directions in any computable fashion, e.g., periodically (descendants of periodic strings are periodic and ultimately give a periodic sequence of generations). For such an effectively infinite m we have corresponding effectively infinite choice of n, for shortening of the daughter of a finite string by one symbol compared to the parent entailed by the daughter algorithm is no longer operative; and only this limits n for a quasigroup.

To see that effectively unbounded k is also obtainable, simply take k as large as desired, partition the k symbols into as many classes as the number of distinct symbols actually needed, and in the final decoding assign all the symbols of a class to the single symbol to which they correspond. Unbounded k can also be obtained with a small alphabet, e.g., binary as mentioned earlier, by encoding the k symbols as k classes of binary strings of desired length. Within a class one can introduce regularities in an encoded message which do not appear in the decoded and translated string. These can be used to combat noise in conventional fashion and/or to lead the enemy to spurious deciphering of the string.

Even the preceding hardly scratches the full gamut of coding procedures available, for the foregoing processes generate sets (sequences) of strings which need not be the actual encoded messages themselves. These can be any sequences of selections of elements from a <u>set</u> of strings from which that set can be generated algorithmically. The message could have been chosen as another such sequence. It would determine the set of strings, and thence the encoding,

and vice versa. This procedure can clearly be iterated; the encoded message determines a set of strings and thus a string, which in turn determines a set and then a string and so on for as many stages as desired to reach the clear string. Furthermore, this can be done in "blocks" in many ways. The received string is partitioned into p parts, $s_1\ s_2 \ldots s_p$, say, s_1 decoded into t_1, the concatenation of t_1 and s_2 decoded into t_2, and so on, $t_{p-1}\ s_p$ being decoded into the clear string t_p. There is no end to such string games. Their number explodes combinatorially like L_k or worse and using the freedom to choose L_k puts L_k in the exponent of the number of such string games. We close this section by showing one simple way to get such a set of strings, and how the use of quasigroups gives unique decodings.

Consider a clear message of length n+1; its daughter will have length n, the n^{th} generation string is one symbol long. View this set of strings as displayed on a triangular grid; the parent symbols are on lattice points in a line, the daughter string on the line below, each symbol forming a little triangle with the symbols of the parent of which it is the product. Successive generations fill out the rest of a large triangle of side (n+1). Choosing any (n+1) symbols, one from each generation, now determines the entire triangle (which contains (n+1)(n+2)/2 symbols). Proof: the last two rows contain 3 points and 2 symbols; the missing one is determined by the multiplication table. This situation recurs in the next row, symbol by symbol, until it is filled, and so on until the whole triangle is obtained. There are clearly (n+1)! ways to choose such a set, and each side can be considered the "parent" of a set of rows. So $(3(n+1)!)^2$ input-output pairs result, and even this does not exhaust the number of such pairs, for as can easily be verified, the 3 vertices of a triangle of side 3 determine the three missing elements, and this case admits systematic extension. The point is that the number of uniquely decipherable encoding systems is so vast that the enemy is killed by

the combinatorics by a mix of simple devices. In the next section we show how bus automata can accomplish parallel decoding of quasigroup algorithms.

III. Bus Automata as Parallel Coders and Decoders

Shannon's work factor concept [6], which encapsulates the essence of the security wall provided by a cipher, has the following practical problem. Complexities introduced to foil an enemy can entail unacceptable costs to the user because of errors, loss of keys, and excessive encoding/decoding times. Computers essentially eliminate errors and are not forgetful. The effect of having highly complex key spaces, but without undue burden on the user, is obtained by combining "user controlled cryptographic transformations" and "system controlled cryptographic transformations", as shown by Gudes et al. [7]. It thus remains to be shown that the enormous key spaces and work factors (tantamount to practically perfect security) which the foregoing formalism provides, can be obtained without excessive computation time penalty to the user.

We have shown in II that computation of a daughter string is parallel computable in one operation by a long shift register or cellular automaton; the decoding process needs more analysis. In all cases mentioned it depends on how fast a parent string can be reconstructed from its daughter, given a single element of the parent string. The search for a method to accomplish this in one parallel operation, rather than by sequential element-by-element computation led the writer to invent bus automata (BA) [1], a generalization of both cellular automata [8] and iterative switching networks or arrays [9]. In brief, the state of a cell comprises both switch settings under local control in each cell (permitting cooperative build-up of communication busses or channels between widely separated cells) and the states of the controlling local automata. Information can then be rapidly exchanged between cells and computations sped by parallel operations in many cells.

In joint work with former students, BA's have been applied to pattern recognition [10] and parallel acceptance of formal languages [11]. They are used to model some aspects of skill acquisition in a forthcoming paper [12], and are proving their value for parallel speed-up statistical data reduction and statistical pattern recognition as well [13]. Returning to the problem at hand, computation of a parent string in a single parallel operation was the first non-obvious successful case, and the foundation of all subsequent work [1]. We now give the parallel parent string algorithm.

By (2.4), the element d_i of the daughter is given by

$$d_i = p_i * p_{i+1}$$

We know the set d_i and one element of the parent, say p_o. As we are dealing with a quasigroup, we can write

$$p_{i+1} = d_i \bar{*} p_i \tag{3.1}$$

where $\bar{*}$ is a new quasigroup operation defined in terms of * by unique solvability. Obviously (3.1) permits consecutive computation of all p_i, i = 1, 2, ..., given the d_i and p_o.

We mention, for its obvious interest and relevance, that quasigroups of all orders exist for which the operations * and $\bar{*}$ are identical; we call them self-solving quasigroups (SSQ) and have proved the following theorems (not yet published), among others. If the multiplication table of any group is modified by replacing every element by its inverse, the multiplication table of an SSQ is obtained. If a finite SSQ is a group, it is the direct product of cyclic groups of order 2 and conversely. All (unipotent or idempotent) quasigroups which correspond to Steiner triple systems (Steiner quasigroups) [14] are SSQ's. The number of SSQ's of any order thus can not be less, and is generally much greater, than the number of groups of that order. Coding/decoding algorithms can be made identical for SSQ's; by interlacing them with transformations which are their own inverses one obtains arbitrarily complex "self-decoding" algorithms.

To solve (3.1) in parallel under the conditions stated, a one-dimensional BA or shift register suffices. If the quasigroup has order k, there will be precisely k ordered pairs of elements whose $\overline{*}$ product is equal to any given element. Indeed we can write

$$d_i = p_i^j * p_{i+1}^j \tag{3.2}$$

$$p_{i+1}^j = d_i \;\overline{*}\; p_i^j \tag{3.3}$$

where j ranges over the k values needed to designate all such ordered pairs. Endow each cell i with k parallel bus elements each of which can connect in series with any one of the k bus elements of cell i+1. Each bus element corresponds to one of the k elements of the quasigroup. When cell i stores element d_i, the cell state connects the k bus elements of cell i individually to elements of cell i+1 so that the k pairs satisfying (3.2), and only those pairs, are connected in series. For the cell containing d_o, only the bus element corresponding to p_o is connected to a signal source; all candidates for p_n, the last symbol of the parent string, have their bus element connected to a signal receiver. The unique parent string possible under the conditions described then permits signal propagation to the receiver precisely through bus elements corresponding to the elements of that unique parent string. The receiver can then actuate parallel output channels from the cells by means of which further parallel processing of the parent string can be accomplished.

To sum up, both parent and daughter algorithms admit parallel processing, and the complex "triangle" processes described earlier use nothing but successions of those algorithms, perhaps with alternation between kinds, and selection of "template" elements. It is readily verified that if one counts a parallel string process as a single operation, then both encoding and decoding processes for an n-character string take n operations (plus element selections if needed). The art of chip microprocessor design has progressed to where real devices embodying these algorithms are within the state of the art. User time cost

grows "linearly" with coding complexity, but the work factor for the enemy grows "combinatorially". Coding algorithms can be easily cascaded to yield more complex algorithms leading to easy updating in the form of new codes.

IV. Conclusion

It seems hard to avoid the conclusion that in a very meaningful sense one can say that the problem of cryptographic security has now been solved in principle. By this we do not mean that a code has been found which could never be broken in principle, but rather that a class of codes has been found for which the labor is too large, even with super-computers (no matter how super), for any code breaker to tolerate. The burden on the user is reasonable, and non-cryptographically motivated coding is compatible, to the point of being integratable with, the coding systems discussed.

REFERENCES

[1] J. Rothstein, Patterns and Algorithms, Proc. IEEE Symp. Adaptive Proc. (9th), Paper II.4 (1970).

[2] J. Rothstein, On the Ultimate Limitations of Parallel Processing, pp. 206-212, Proc. 1976 International Conf. on Parallel Processing, Detroit (1976).

[3] M. Hall, Distinct Representatives of Subsets, Bull. Amer. Math. Soc. 51 922-6 (1945).

[4] S. E. Bammel and J. Rothstein, Discrete Mathematics 11, 93-5 (1975).

[5] H. J. Ryser, Permanents and Systems of Distinct Representatives, in Combinatorial Mathematics and its Applications, pp. 55-8 Univ. of N. Carolina Press (Chapel Hill, 1969).

[6] C. E. Shannon, Communication Theory of Secrecy Systems, Bell Syst. Tech. Jour. 28,656-715 (1949).

[7]a E. Gudes, The Application of Cryptography to Data Base Security, Ph.D. dissertation, The Ohio State University, 1976, (J. Rothstein, advisor).

[7]b E. Gudes, F. A. Stahl, H. S. Koch, Applications of Cryptographic Transformations to Data Base Security, NCC Proceedings, 1976.

[7]c E. Gudes and J. Rothstein, papers in progress.

[8] A few standard references on cellular automata are

(a) J. von Neumann, Theory of Self-Reproducing Automata, University of Illinois Press (Urbana 1966).
(b) Essays on Cellular Automata, A. W. Burks, ed., University of Illinois Press (Urbana 1968).
(c) E. F. Codd, Cellular Automata, Academic Press (New York 1968).

[9] R. C. Minnick, A Survey of Microcellular Research, Jour. Assoc. Comp. Mach. 14, 203-241 (1967).

[10] J. Rothstein and CFR Weiman, Parallel and Sequential Specification of a Context Sensitive Language for Straight Lines on Grids, Computer Graphics and Image Processing 5, 106-124 (1976)

[11] Joint papers of the writer and J. M. Moshell in preparation; see Ph.D. dissertation of the latter, Parallel Recognition of Formal Languages by Cellular Automata, The Ohio State University, 1975, and their joint conference papers and technical reports.

(a) Parallel Recognition of Patterns: Insights from Formal Language Theory, pp. 222-229, Proc. 1976 Int. Conf. on Parallel Processing (Detroit 1976).
(b) Bus Automata, Tech. Rpt. CS-76-14, Computer Science Department University of Tennessee, 1976.

[12] J. Rothstein, Transitive Closure, Parallelism, and the Modeling of Skill Acquisition, Paper accepted for the 1977 Int'l. Conf. on Cybernetics and Society, Washington, D. C., Sept. 19-21 (1977).

[13] Dissertation work of John Mellby (J. Rothstein, advisor) in progress at Ohio State University.

[14] J. Denes and A. D. Keedwell, Latin Squares and Their Applications, Academic Press (New York, 1974).